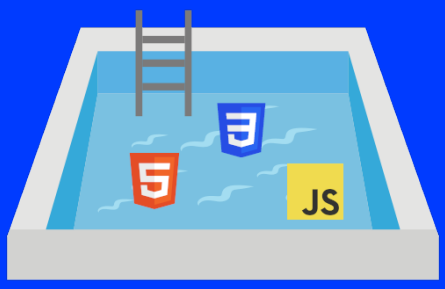
{EPITECH}

# WEB DEV POOL_

< DAY 09 - PHP />



# WEB DEV POOL

## Before you start

Now that you now the basics of programming in PHP, let's use it to interact with HTML and forms.

### Setup a web server

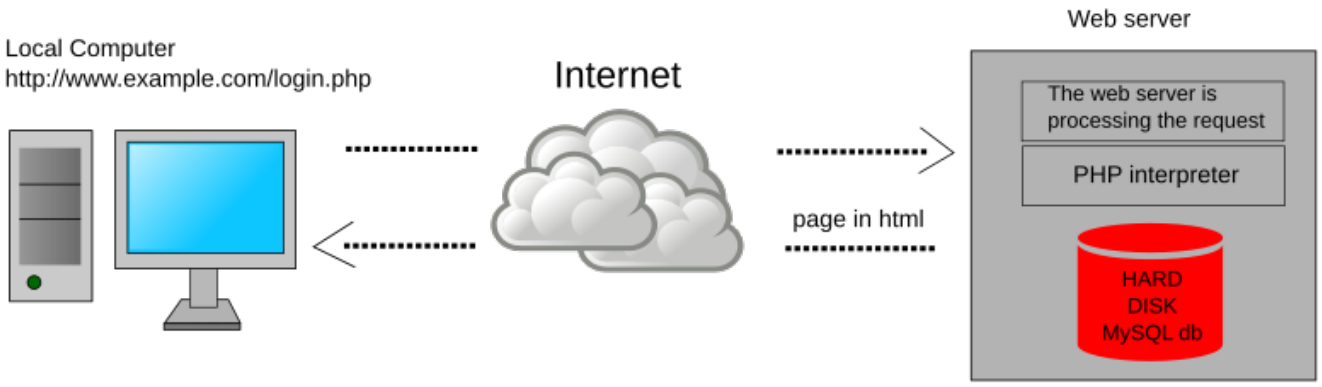
Contrary to HTML, CSS and Javascript which are interpreted client-side by your web browser, PHP is a back-end language.

It needs a server that will interpret the language (as does the `php` command you've used yesterday) when a URL is called from the browser.

We won't detail it here but it would be useful for you to install and configure a simple PHP web server for testing purposes.

Here's a tutorial: How to Install LAMP, but you can find many more on the Internet.

Apache and Nginx are both great web servers capable of processing PHP.



## Task 01

**Delivery**: `./task01.php`

Create a `display_menu` function that:

- ✓ takes no parameters ;
- ✓ returns a string containing HTML capable of rendering a menu, like this:

```
<ul>
    <li><a href="home.php">Home</a></li>
    <li><a href="product.php">Products</a></li>
    <li><a href="about.php">About Us</a></li>
    <li><a href="contact.php">Contact</a></li>
</ul>
```

Because you've setup a web server, you can test the function.

See how your browser renders the menu by creating a page like `index01.php` containing:

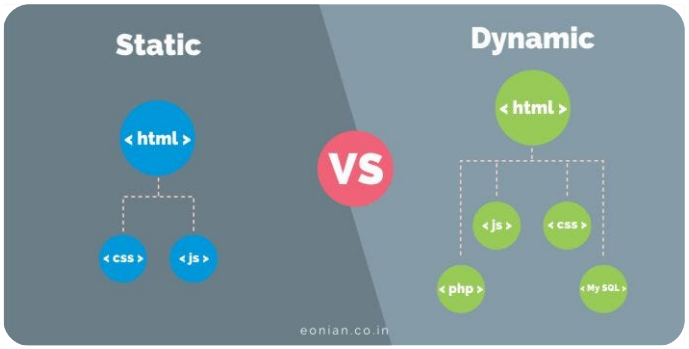
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Task 01</title>
</head>
<body>
    <?php
        require("task01.php");
        $menu = display_menu();
        echo $menu;
    ?>
</body>
</html>
```

Then, go to `http://127.0.0.1/index01.php` where you SHOULD see your page with the menu.

## Task 02

**Delivery**: `./task02.php`

Remembers day01 of this pool?
If you keep just the content of the `<body>` of each page in separate HTML files, such as `home.html`, `php.html` and `sql.html`, you would be able to **include** it in your structure.
PHP can be used to render **dynamic content** without having to re-write all of the page structure for each new page of your website.




Create a `render_body` function which takes a string as parameter.
If the string is `home`, `php` or `sql`, returns the content of the corresponding HTML file.
If the parameter is unknown, just return: `<p>Unknown page</p>`.

The corresponding HTML files must be present in the same directory as `task02.php`

Test the function: see if your browser render the body of a page like `index02.php` containing:

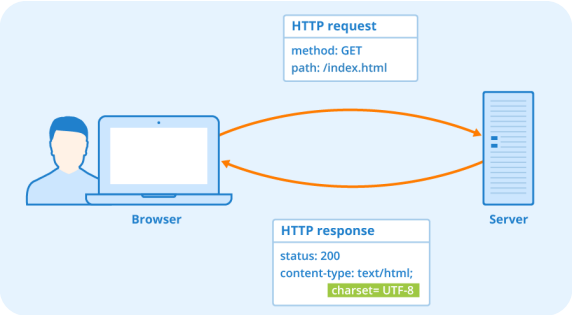
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Task 02</title>
</head>
<body>
    <?php
        require("task02.php");
        $body = render_body("home");
        echo $body;
    ?>
</body>
</html>
```

Go to `http://127.0.0.1/index02.php`.
You should see your page with the content of `home.html` inside the `<body>` tag.
If you change the value of the parameter, the content of the page changes.

## Task 03

**Delivery**: `./task03.php`

The previous task was good, but not enough.
Indeed, it's not fully dynamic as you edit your index file to change a parameter value.
We'll now become truly dynamic by using *URL parameters* (such as **GET**).




Create a `dynamic_body` function which takes no parameter. It'll check the value of the GET parameter called `page`. Depending on the value of this `page` parameter, do the same thing as the previous task. If there is no `page` parameter or if its value is unknown, just return: `<p>Unknown page</p>`.

There's a special variable in PHP called `$_GET`.

Test the function: see if your browser render the body of a page `index03.php` containing:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Task 03</title>
</head>
<body>
    <?php
        require("task03.php");
        $body = dynamic_body();
        echo $body;
    ?>
</body>
</html>
```

Go to `http://127.0.0.1/index03.php?page=home`.
You should see your page with the content of `home.html` inside the `<body>` tag.
You can now change the value of `page` in the URL to change the content of the body.
That's what I call dynamic, you no longer have to edit your file.
All your web pages now shares the same structure using a single file.

## Task 04

**Delivery**: `./task04.php`

Now that you know how to handle URL parameters with `$_GET` we'll see another way of sending data: POST. GET and POST are different in terms of limitations and usage. POST is most commonly used to handle forms data.

More information on GET vs POST and example here on w3schools.

Create a function `whoami` which takes no parameters and prints "Hi, my name is <name> and I'm <age> years old.". The `name` and `age` parameters will come as **POST** data.
If there's no name, print: "Hi, I have no name and I'm <age> years old."
If there's no age or it's not valid, print: "Hi, my name is <name>."
If there's neither a name or a valid age, guess what you should print.

If you've setup a web server, you can test the function using the `curl` command.

**Examples:**

```
Terminal
$> cat index04.php
<?php
require("task04.php");
whoami();
?>
```

```
Terminal
$> curl -d "name=Jane&age=21" -H "Content-Type: application/x-www-form-urlencoded" -X POST
http://127.0.0.1/index04.php
Hi, my name is Jane and I'm 21 years old.
```

```
Terminal
$> curl -d "nom=John&age=48" -H "Content-Type: application/x-www-form-urlencoded" -X POST
http://127.0.0.1/index04.php
Hi, I have no name and I'm 48 years old.
```

## Task 05

**Delivery**: `./task05.php`

We'll now connect our PHP script to a real HTML form. You have to make the following page work (you can name it `index5.php` but you don't have to turn it in!).

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Task 05</title>
</head>
<body>
  <?php
   require("task05.php");
   if (form_is_submitted()) {
   ?>
  <p><?php whoami(); ?></p>
  <?php } else { ?>
  <form method="post">
    <div>
      <label for="name">Name</label>
      <input type="text" id="name" name="name" />
    </div>
    <div>
      <label for="age">Age</label>
      <input type="number" id="age" name="age" min="0" />
    </div>
    <div>
      <label for="curriculum">Curriculum</label>
      <select name="curriculum" id="curriculum">
        <option value="">--Please choose an option--</option>
        <option value="pge">PGE (Programme Grande Ecole)</option>
        <option value="msc">MSc Pro</option>
        <option value="coding">Coding Academy</option>
        <option value="wac">Web@cademie</option>
      </select>
    </div>
    <div>
      <input type="submit" name="submit" value="Send" />
    </div>
  </form>
  <?php } ?>
</body>
</html>
```

Thus, you have to create two functions:

- ✓ `form_is_submitted` returns a boolean value (form has already been submitted or not).
- ✓ `whoami` works as in the previous task with an added feature. It must add "I'm a student of <curriculum>." if there's a curriculum specified after the first sentence. Example: "Hi, I have no name and I'm 48 years old. I'm a student of MSC Pro."

v 3.4.1